UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/684,053	10/09/2003	Chandan Mathur	1934-12-3	3240

7590 07/19/2007

Bryan A. Santarelli
GRAYBEAL JACKSON HALEY LLP
Suite 350
155-108th Avenue NE
Bellevue, WA 98004-5901

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
07/19/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/684,053	MATHUR ET AL.	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 May 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-61* is/are pending in the application.

4a) Of the above claim(s) *25-36 and 55-61* is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24 and 37-54* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *14 May 2004* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *4/11/04, 11/8/06, 2/15/07*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-61 are pending. Claims 25-36 and 55-61 have been withdrawn. Claims 1-24 and 37-54 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: IDS as received on 11/8/2006, IDS and Non-compliant Amendment as received on 1/25/2007, IDS as received on 2/15/2007, and Amendment as received on 5/7/2007.

Specification

3. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

Drawings

4. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference character(s) not mentioned in the description:

- In the specification amendment to paragraph [50], applicant deleted reference number 58, which is in Fig.3. Consequently, reference number 58 no longer appears in the specification, as the deleted reference was the only reference to that number.

Corrected drawing sheets in compliance with 37 CFR 1.121(d), or amendment to the specification to add the reference character(s) in the description in compliance with 37 CFR

1.121(b) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1-3, 5-6, 9-12, and 14-17 are rejected under 35 U.S.C. 102(b) as being anticipated by Tamura, U.S. Patent No. 6,108,693.

7. Referring to claim 1, Tamura has taught a computing machine, comprising:

a) a first buffer. See Fig.4, component 442a, and note the first communication buffer.

b) a processor (Fig.4, components 410/420 form a multiprocessor) coupled to the buffer and operable to:

b1) execute an application, a first data-transfer object, and a second data-transfer object.

A processor executes an application, where an application is any program that causes

operations to be performed by the processor. Also, first and second data transfer objects are executed. See Fig.4, components 400 and 430, respectively.

b2) publish data under the control of the application. See column 8, lines 63-64, and note that data of array A is published (i.e., generated).

b3) load the published data into the buffer under the control of the first data-transfer object. See Fig.6, and note the first data transfer object (transmit object) loads the published data into the buffer.

b4) retrieve the published data from the buffer under the control of the second data-transfer object. See Fig.6, and note that the second data transfer object (receive object) retrieves the data from the buffer.

8. Referring to claim 2, Tamura has taught a computing machine as described in claim 1. Tamura has further taught that the first and second data-transfer objects respectively comprise first and second instances of the same object code. From Fig.6, it can be seen that the first object writes data to the buffer and the second object reads data from the buffer. Both objects clearly access the buffer, and consequently, both comprise code to access the buffer. The objects' code is the same in at least this respect (it causes buffer access).

9. Referring to claim 3, Tamura has taught a computing machine as described in claim 1. Tamura has further taught the processor comprises:

a) a processing unit operable to execute the application and publish the data under the control of the application. Recall from the rejection of claim 1 that the processor executes an application and publishes data under control of the application. This execution and publishing is performed by a processing unit.

b) a data-transfer handler operable to execute the first and second data-transfer objects, to load the published data into the buffer under the control of the first data-transfer object, and to retrieve the published data under the control of the second data-transfer object. Again, recall from the rejection of claim 1 that first and second objects, for loading and retrieving from a buffer, respectively, are executed. This unit which performs this execution, loading, and retrieving is a data transfer handler.

10. Referring to claim 5, Tamura has taught a computing machine as described in claim 1. Tamura has further taught that the processor is further operable to:

a) execute a queue object and a reader object. See column 2, lines 56-67, and note that the queue object writes to the buffer and sets the buffer from read-disabled to read-enabled. The reader object sets the buffer from write-disabled to write-enabled.

b) store a queue value under the control of the queue object, the queue value reflecting the loading of the published data into the buffer. Again, see column 2, lines 56-67, and note that by storing a value indicative of the buffer being read-enabled in flag 442b (Fig.4), the system is signaling that published data has been loaded into the buffer and it can now be read.

c) read the queue value under the control of the reader object. See column 8, lines 27-29. The system must first detect whether the buffer is read-enabled, and therefore it must read the queue value. If it is read-enabled, then the buffer can be read.

d) notify the second data-transfer object that the published data occupies the buffer under the control of the reader object and in response to the queue value. See column 8, lines 29-31. If the read enable flag is set, then the published data may be read by the second data transfer object.

e) retrieve the published data from the buffer under the control of the second data-transfer object and in response to the notification. See Fig.1, component 23, and Fig.4, component 430, and Fig.6.

11. Referring to claim 6, Tamura has taught a computing machine as described in claim 1. Tamura has further taught:

a) a bus. See Fig.4 and note the bus between local memory and a single processor of the multiprocessor.

b) wherein the processor is operable to execute a communication object and to drive the retrieved data onto the bus under the control of the communication object. The abstract discloses that data is retrieved from the buffer and sent to the local memory via a bus. The object which causes this to happen is the communication object.

12. Referring to claim 9, Tamura has taught a computing machine as described in claim 1. Tamura has further taught that:

a) the first and second data-transfer objects respectively comprise first and second instances of the same object code. From Fig.6, it can be seen that the first object writes data to the buffer and the second object reads data from the buffer. Both clearly access the buffer, and consequently, both comprise code to access the buffer. The code of both objects is the same in at least this respect (it causes buffer access).

b) the processor is operable to execute an object factory and to generate the object code under the control of the object factory. All processors execute programs. The program (object factory) will dictate when data needs to be transmitted and received. That is, when the program calls for

data to be transmitted, the first data transfer object will be invoked so that data may be transmitted.

13. Referring to claim 10, Tamura has taught a computing machine, comprising:

a) a first buffer. See Fig.4, component 442a, and note the first communication buffer.

b) a processor (Fig.4, components 410/420 form a multiprocessor) coupled to the buffer and operable to:

b1) execute first and second data-transfer objects and an application. A processor executes an application, where an application is any program that causes operations to be performed by the processor. Also, first and second data transfer objects are executed. See Fig.4, components 400 and 430, respectively.

b2) retrieve data and load the retrieved data into the buffer under the control of the first data-transfer object. See the abstract and Fig.6, and note the first data transfer object (transmit object) retrieves data from local memory and loads the data into the buffer.

b3) unload the data from the buffer under the control of the second data-transfer object. See Fig.6, and note that the second data transfer object (receive object) retrieves the data from the buffer.

b4) process the unloaded data under the control of the application. After the data is retrieved from the buffer, it is processed by storing it in local memory. See the abstract.

14. Referring to claim 11, Tamura has taught a computing machine as described in claim 10. Tamura has further taught that the first and second data-transfer objects respectively comprise first and second instances of the same object code. From Fig.6, it can be seen that the first object writes data to the buffer and the second object reads data from the buffer. Both clearly access

the buffer, and consequently, both comprise code to access the buffer (the code is the same in at least this respect).

15. Referring to claim 12, Tamura has taught a computing machine as described in claim 10. Tamura has further taught that the processor comprises:

a) a processing unit operable to execute the application and process the unloaded data under the control of the application. Recall from the rejection of claim 10 that the processor executes an application and processes unloaded data under control of the application. This execution and processing is performed by a processing unit.

b) a data-transfer handler operable to execute the first and second data-transfer objects, to retrieve the data from the bus and load the data into the buffer under the control of the first data-transfer object, and to unload the data from the buffer under the control of the second data-transfer object. Again, recall from the rejection of claim 10 that first and second objects, for loading and unloading from a buffer, respectively, are executed. This unit which performs this execution, loading, and unloading is a data transfer handler.

16. Referring to claim 14, Tamura has taught a computing machine as described in claim 10. Tamura has further taught that the processor is further operable to:

a) execute a queue object and a reader object. See column 2, lines 56-67, and note that the queue object writes to the buffer and sets the buffer from read-disabled to read-enabled. The reader object sets the buffer from write-disabled to write-enabled.

b) store a queue value under the control of the queue object, the queue value reflecting the loading of the retrieved data into the first buffer. Again, see column 2, lines 56-67, and note that

by storing a value indicative of the buffer being read-enabled in flag 442b (Fig.4), the system is signaling that published data has been loaded into the buffer and it can now be read).

c) read the queue value under the control of the reader object. See column 8, lines 27-29. The system must first detect whether the buffer is read-enabled, and therefore it must read the queue value. If it is read-enabled, then the buffer can be read.

d) notify the second data-transfer object that the retrieved data occupies the buffer under the control of the reader object and in response to the queue value. See column 8, lines 29-31. If the read enable flag is set, then the published data may be read.

e) unload the retrieved data from the buffer under the control of the second data-transfer object and in response to the notification. See Fig.1, component 23, and Fig.4, component 430, and Fig.6.

17. Referring to claim 15, Tamura has taught a computing machine as described in claim 10. Tamura has further taught:

a) a second buffer. See Fig.4, component 442a, and Fig.6, and note that a second buffer exists.

b) wherein the processor is operable to retrieve the data from the second buffer under the control of the first data-transfer object. See Fig.6 and note that data is received from the second buffer. And, the receiving is under the control of the first data transfer object because the first object writes the data to the buffer and sets the read-enabled status flag for that buffer so that the receiving portion may read it. See column 2, lines 56-67.

18. Referring to claim 16, Tamura has taught a computing machine as described in claim 10. Tamura has further taught:

a) a bus. See Fig.4 and note the bus between local memory and a single processor of the multiprocessor.

b) wherein the processor is operable to execute a communication object, to receive the data from the bus under the control of the communication object, and to retrieve the data from the communication object under the control of the first data-transfer object. The object which causes sending and retrieving of data on a bus (via a buffer) is the communication object. The communication object will first receive data from local memory via a bus (see the abstract and Fig.4, and note the bus coupling local memory to component 410), and after the data is written to the buffer it is retrieved. See Fig.6. The retrieval is under control of the communication object as the communication object will set the read-enable flag which allows retrieval to occur. See column 2, lines 56-67.

19. Referring to claim 17, Tamura has taught a computing machine as described in claim 10. Tamura has further taught that:

a) the first and second data-transfer objects respectively comprise first and second instances of the same object code. From Fig.6, it can be seen that the first object writes data to the buffer and the second object reads data from the buffer. Both, however, clearly access the buffer, and consequently, both comprise code to access the buffer. The code is the same in at least this respect (it causes buffer access).

b) the processor is operable to execute an object factory and to generate the object code under the control of the object factory. All processors execute programs. The program will dictate when data needs to be transmitted and received. That is, when the program calls for data to be transmitted, the first data transfer object will be invoked so that data may be transmitted.

20. Claims 37, 39-43, 45, and 47-49 are rejected under 35 U.S.C. 102(b) as being anticipated by Nakagoshi et al., U.S. Patent No. 5,377,333 (herein referred to as Nakagoshi).

21. Referring to claim 37, Nakagoshi has taught a method comprising:

a) publishing data with an application. See Fig.1 and note that data is passed among processing elements. This data is inherently published (generated) by some application.

b) loading the published data into a first buffer with a first data-transfer object. See column 7, line 62, to column 8, line 43, discuss that each element in the array is able to transfer data to another element in the array, where data may be received and sent by intermediate nodes on the way. Also, see Fig.9 and the description of Fig.9 in column 11, lines 1-24. Published data, when transferred, will be loaded into a first buffer (FIFO) in an exchanger 102, where the loading is inherently performed by some logic (first data-transfer object).

c) retrieving the published data from the buffer with a second data-transfer object. See Fig.9 and column 11, lines 1-24. Note that after being stored in an exchanger, the data will be retrieved by additional logic (at least the routing control 902) within the exchanger. The routing means determines if the data needs to be routed further to another node or if it has reached its destination.

d) generating a message header that includes a destination of the retrieved data. See Fig.5, component 501.

e) generating a message that includes the retrieved data and the message header. If the data is not at its final destination, then the data and header are sent to the next appropriate node.

22. Referring to claim 39, Nakagoshi has taught a method as described in claim 37. Nakagoshi has further taught:

a) generating a queue value that corresponds to the presence of the published data in the buffer. See column 11, lines 19-24. The empty value indicates when data is in the queue/buffer.

b) notifying the second data-transfer object that the published data occupies the buffer in response to the queue value. See column 11, lines 19-24. If the empty flag is clear, this indicates that the buffer may be read, at which point a read strobe signal is produced.

c) wherein retrieving the published data comprises retrieving the published data from the buffer with the second data-transfer object in response to the notification. See column 11, lines 19-24.

23. Referring to claim 40, Nakagoshi has taught a method as described in claim 37. Nakagoshi has further taught driving the retrieved data onto a bus with a communication object. From the cited passages and at least Fig.1, it should be realized that data is passed between a node A and a node B, where if node A and node B are more than one node apart, at least a third node will act as an intermediate node that assists in the transfer. Therefore, node C might receive data from node A and then further drive the same data on a bus to node B. The inherently existing object which causes this to happen is the communication object.

24. Referring to claim 41, Nakagoshi has taught a method as described in claim 37. Nakagoshi has further taught loading the retrieved data into a second buffer with the second data-transfer object. From the cited passages and at least Fig.1, it should be realized that data is passed between a node A and a node B, where if node A and node B are more than one node apart, at least a third node will act as an intermediate node that assists in the transfer. Therefore, node C might receive data from node A and then further drive the same data on a bus to node B,

where it is loaded into node B's buffer (second buffer). The inherently existing logic which causes this loading to happen is at least part of the second data-transfer object.

25. Referring to claim 42, Nakagoshi has taught a method as described in claim 37. Nakagoshi has further taught that generating the message header and the message comprise generating the message header and the message with the second data transfer object. Clearly, a "second data transfer object" is very broad. As long as the system performs the above generation, which Nakagoshi does, then it can be said that the portion which does the generating is at least part of the second data transfer object.

26. Referring to claim 43, Nakagoshi has taught a method as described in claim 37. Nakagoshi has further taught:

a) generating data-transfer object code with an object factory. All processors execute programs. The program (object factory) will dictate/generate when data needs to be transmitted and received. That is, when the program calls for data to be transmitted or received, the data transfer objects will be invoked so that data may be transmitted and received. Clearly, the hardware shown in the figures must have some software interaction because without software, the hardware would be useless.

b) generating the first data-transfer object as a first instance of the object code. When data needs to be transferred, some type of "send" or "pass" or "store" command must be generated. This command is part of the data transfer object.

c) generating the second data-transfer object as a second instance of the object code. Similarly, when data is to be read from the buffer and brought into the system, some type of command needs to be generated. This command is part of the second data transfer object.

27. Referring to claim 45, Nakagoshi has taught a method comprising:

a) receiving a message that includes data and that includes a message header that indicates a destination of the data. See Fig.1, Fig.5, Fig.9, and column 11, lines 1-24, and column 7, line 39, to column 8, line 43. Basically, nodes send message to one another, through intermediate nodes. Each node includes an exchanger, as shown in Fig.9. When a message arrives at a node, it is received as a message including a header and data (Fig.5).

b) loading the received data into a first buffer with a first data-transfer object, the first buffer corresponding to the destination. See Fig.9, and column 11, lines 1-24. Data that is received at a node is loaded into a buffer (FIFO).

c) unloading the data from the buffer with a second data-transfer object. See Fig.9, and column 11, lines 1-24, and note that data is then unladed from the buffer either for processing (if the data has reached its final destination) or for further routing (if the data has not yet reached its final destination).

d) processing the unloaded data with an application corresponding to the destination. As mentioned above, a processor which receives the data will process the data in some form. Otherwise, there'd be no point in passing data. And, the processor to do the processing will be the one that the message was passed to (the destination).

28. Referring to claim 47, Nakagoshi has taught a method as described in claim 45. Nakagoshi has further taught:

a) generating a queue value that corresponds to the presence of the data in the buffer. See column 11, lines 19-24. The empty value indicates when data is in the queue/buffer.

b) notifying the second data-transfer object that the data occupies the buffer in response to the queue value. See column 11, lines 19-24. If the empty flag is clear, this indicates that the buffer may be read, at which point a read strobe signal is produced.

c) wherein unloading the data comprises unloading the data from the buffer with the first data-transfer object in response to the notification. See column 11, lines 19-24.

29. Referring to claim 48, Nakagoshi has taught a method as describe in claim 45. Nakagoshi has further taught receiving the message comprises receiving the message with the first data-transfer object. Clearly, as claimed, a "first data transfer object" is very broad. As long as the system performs the above receiving, which Nakagoshi does, then it can be said that the portion which does the receiving is at least part of the first data transfer object.

30. Referring to claim 49, Nakagoshi has taught a method as described in claim 45. Nakagoshi has further taught:

a) receiving the message comprises retrieving the message from a bus with a communication object. From the cited passages and at least Fig.1, it should be realized that data is passed and received from a communication bus and stored within processing element buffers. The inherently existing object which causes this receiving to happen is the communication object.

b) transferring the data from the communication object to the first data transfer object. See Fig.9 and column 11, lines 1-24. Data may be received at one node (via a communication object) and then sent to another nodes. Therefore, the logic further sending the data is the first data transfer object. Note that for it to be able to send data it must receive the data from the communication object.

Claim Rejections - 35 USC § 103

31. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

32. Claims 4 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura.

33. Referring to claim 4, Tamura has taught a computing machine as described in claim 1. Tamura has not taught that the processor is further operable to execute a thread of the application and to publish the data under the control of the thread. However, Official Notice is taken that multithreaded processors and their advantages are well known and accepted in the art. Specifically, it is known to divide up a program into threads in order to increase efficiency by reducing stall time. With multiple threads, the system may switch to a second thread when a first thread stalls, thereby hiding the stall time require by the first thread. Essentially, the processor is kept busy as often as possible with multithreading. As a result, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the processor executes a thread of the application and publishes data under control of the thread.

34. Referring to claim 13, Tamura has taught a computing machine as described in claim 10. Tamura has not taught that the processor is further operable to execute a thread of the application and to process the unloaded data under the control of the thread. However, Official Notice is taken that multithreaded processors and their advantages are well known and accepted in the art. Specifically, it is known to divide up a program into threads in order to increase efficiency by

reducing stall time. With multiple threads, the system may switch to a second thread when a first thread stalls, thereby hiding the stall time require by the first thread. Essentially, the processor is kept busy as often as possible with multithreading. As a result, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the processor executes a thread of the application and processes the unloaded data under control of the thread.

35. Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura in view of Mishler, U.S. Patent No. 5,283,883.

36. Referring to claim 7, Tamura has taught a computing machine as described in claim 1. Tamura has not taught a second buffer and that the processor is operable to provide the retrieved data to the second buffer under the control of the second data-transfer object. However, Mishler has taught the concept of a DMA unit having a buffer that holds data that is to be written to memory. See Fig.6, component 280, for instance. A DMA unit is used to provide very fast data transfers between a processor and memory since the DMA unit can transfer the data without full CPU involvement (thereby allowing the CPU to perform other important tasks). See column 1, lines 28-39. Since the data is being transferred from the receiving portion of the multiprocessor to the local memory (see the abstract of Tamura), it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura to include a DMA unit having a buffer (second buffer), which buffers the data received from the first buffer under the control of the second data transfer object. One would be motivated to make such a combination because the DMA unit and its buffer would allow for faster transfer of data from the receiving portion to

the local memory while allowing the CPU to focus on other tasks not involving data transfer to memory.

37. Claims 8 and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura in view of The "Microsoft Computer Dictionary, 4th Edition," Microsoft Press, 1999 (herein referred to as Microsoft).

38. Referring to claim 8, Tamura has taught a computing machine as described in claim 1. While Tamura has further taught that messages are sent between processors (column 1, lines 51-52), Tamura has not explicitly taught that the processor is further operable to generate a message that includes a header and the retrieved data under the control of the second data-transfer object. However, Microsoft has taught that a header is an information structure that precedes and identifies the information that follows, such as a block of bytes in communications (i.e., body of a message). Headers are common in message passing and they typically include parity bits (or other error detection bits) to ensure that the data received is error-free, and the length of the data that follows so that it can be detected whether or not the entire message has been received. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura in view of Morikawa to attach a header, as taught by Microsoft, to the data sent in order to detect errors and length of the data. And, the second object, by retrieving the message, generates the message for additional processing on the receiving end.

39. Referring to claim 18, Tamura has taught a computing machine as described in claim 10. While Tamura has further taught that messages are sent between processors (column 1, lines 51-52), Tamura has not explicitly taught that the processor is further operable to recover the data

from a message that includes a header and the data under the control of the first data-transfer object. However, Microsoft has taught that a header is an information structure that precedes and identifies the information that follows, such as a block of bytes in communications. Headers are common in message passing and they typically include parity bits (or other error detection bits), to ensure that the data received is error-free, and the length of the data that follows so that it can be detected whether or not the entire message has been received. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura to attach a header, as taught by Microsoft, to the data sent in order to detect errors and length of the data. And, clearly, the header must be attached before the data is sent so it would be under control of the first data transfer object.

40. Claims 19-24 and 51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura in view of Morikawa et al., U.S. Patent No. 5,909,565 (herein referred to as Morikawa).

41. Referring to claim 19, Tamura has taught a peer-vector machine comprising:

a) a buffer. See Fig.4, component 442a, and Fig.6.

b) a bus. See Fig.4 and Fig.6 and note that data must travel on a bus, otherwise it cannot be sent anywhere.

c) a processor (Fig.4, components 410/420 form a multiprocessor) coupled to the buffer and to the bus and operable to:

c1) execute an application, first and second data-transfer objects, and a communication object. A processor executes an application, where an application is any program that causes operations to be performed by the processor. Also, first and second data transfer

objects are executed. See Fig.4, components 400 and 430, respectively. The communication object may be interpreted as the overall portion of the program that retrieves data from local memory (abstract, Fig.4), sends the data, receives the data, and sends the received data to another location.

c2) publish data under the control of the application. See column 8, lines 63-64, and note that data of array A is published (i.e., generated).

c3) load the published data into the buffer under the control of the first data-transfer object. See Fig.6, and note the first data transfer object (transmit object (Fig.4, component 400)) loads the published data into the buffer.

c4) retrieve the published data from the buffer under the control of the second data-transfer object. See Fig.6, and note that the second data transfer object (receive object (Fig.4, component 430)) retrieves the data from the buffer.

c5) drive the published data onto the bus under the control of the communication object. As the data is retrieved from a buffer coupled to the system (Fig.6), it is driven onto a bus. Otherwise, data could not be received.

d) Tamura has not taught a pipeline accelerator coupled to the bus and operable to receive the published data from the bus and to process the received published data. However, Morikawa has taught the concept of a pipeline coprocessor (Fig.4, component 202, and the bottom of Fig.5 and Fig.6 show that the coprocessor is pipelined) that receives data from a processor via a buffer, processes the data, and then passes it back via a buffer. As disclosed in Morikawa in column 1, lines 15-19, coprocessors execute special instructions at high speed, and therefore, a processor, which is not especially equipped to handle such instructions at high speed, will pass off

instructions/data to the coprocessor so that it may perform the operation at a more rapid pace. This clearly speeds up the system, and as a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the portion of the system receiving the data is a pipeline accelerator (coprocessor) which performs operations at high speed for the processor.

42. Referring to claim 20, Tamura in view of Morikawa has taught a peer vector machine as described in claim 19.

a) Tamura has further taught that the processor is further operable to construct a message that includes the published data under the control of the second data-transfer object and to drive the message onto the bus under the control of the communication object. Any data that is sent to the buffer is a message. The construction of this message would at least include retrieving it from local memory and outputting it to the buffer. This is done under the control of the second data transfer object because the second object informs the sending portion of the multiprocessor that it may write to the buffer (i.e., it may send the message). See column 2, lines 56-67. And, if data is being passed from point A to point B, then it is being passed via a bus. The driving of the message on the bus is done by the "communication object."

b) Tamura in view of Morikawa has further taught that the pipeline accelerator is operable to receive the message from the bus and to recover the published data from the message. See Fig.4 of Morikawa and again recall that when the message is retrieved by the receiving side, the data is recovered as it is at least part of the message, and this data must be written to the local memory.

43. Referring to claim 21, Tamura in view of Morikawa has taught a peer vector machine as described in claim 19. Tamura in view of Morikawa has further taught:

a) a registry coupled to the processor and operable to store object data. See Fig.4 of Tamura, and note that the objects 400 and 430 are software objects that include instructions. These instructions must be stored somewhere. The storage holding them is the registry.

b) wherein the processor is operable to execute an object factory, and to generate the first and second data-transfer objects and the communication object from the object data under the control of the object factory. All processors execute programs. The program (object factory) will dictate when data needs to be transmitted and received. That is, when the program calls for data to be transmitted, received, and driven onto a bus, the first and second data transfer objects (Tamura, Fig.4, components 400 and 430) will be invoked/generated so that data may be transmitted and the object factory will be generated to that the data may be driven onto a bus.

44. Referring to claim 22, Tamura has taught a peer vector machine comprising:

a) a buffer. See Fig.4, component 442a, and Fig.6 (note the first buffer).

b) a bus. See Fig.4 and note the bus coupling the local memory to the sending portion of the multiprocessor. Also, if data is sent from point A to point B, there must be a bus to carry the data. Consequently, there is a bus coupling transmitting portion 410 (Fig.4) to the communication buffer so that data may be sent as shown in Fig.6. These two buses together is, collectively, "the bus" (i.e., wires that carry data).

c) a portion coupled to the bus and operable to generate data and to drive the data onto the bus. See Fig.4, component 410. This portion generates data (produces data from array A (column 8, lines 63-66) and drives it on the bus from local memory so that it may be received by the transmitting portion 410. Tamura has not explicitly taught that the portion is a pipeline accelerator. However, Morikawa has taught the concept of a pipeline processor (Fig.4,

component 201, and the top of Fig.5 and Fig.6 show that the processor is pipelined) that receives data from a local memory and passes the data to a coprocessor. As is known, pipelining allows for the overlapping and parallelization of instructions, thereby increasing throughput and execution speeds. Consequently, in order to increase efficiency and throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the data is generated and driven onto a bus with a pipeline accelerator, as taught by Morikawa. It should be noted that the pipeline accelerator may be considered to be the portion of the processor which deals with transfer of data. Normal processing would be done by the non-accelerator part.

d) a processor (Fig.4, components 410/420 form a multiprocessor) coupled to the buffer and to the bus and operable to:

d1) execute an application, first and second data-transfer objects, and a communication object. A processor executes an application, where an application is any program that causes operations to be performed by the processor. Also, first and second data transfer objects are executed. See Fig.4, components 400 and 430, respectively. The communication object may be interpreted as the overall portion of the program that retrieves data from local memory (abstract, Fig.4), sends the data, receives the data, and sends the received data to another location.

d2) receive the data from the bus under the control of the communication object. The abstract discloses that data is retrieved from the buffer and sent to the local memory via a bus. The object which causes this to happen is the communication object.

d3) load the received data into the buffer under the control of the first data-transfer object. See Fig.6, and note the first data transfer object (transmit object (Fig.4, component 400)) loads the published data into the buffer.

d4) unload the data from the buffer under the control of the second data-transfer object. See Fig.6, and note that the second data transfer object (receive object (Fig.4, component 430)) retrieves the data from the buffer.

d5) process the unloaded data under the control of the application. After the data is retrieved from the buffer, it is processed by storing it in local memory. See the abstract. Processing is simply performing an operation on data, and storing data is an operation performed on data.

45. Referring to claim 23, Tamura in view of Morikawa has taught a peer vector as described in claim 22.

a) Tamura in view of Morikawa has further taught that the pipeline accelerator is further operable to construct a message that includes the data and to drive the message onto the bus. Any data that is sent to the buffer is a message. The construction of this message would at least include retrieving it from local memory and outputting it to the buffer. And, if data is being passed from point A to point B, then it is being passed via a bus. The driving of the message on the bus is done by the "communication object."

b) the processor is operable to receive the message from the bus under the control of the communication object, and recover the data from the message under the control of the first data-transfer object. . See Fig.4 and Fig.6 of Tamura and again recall that when the message is retrieved by the receiving side of the multiprocessor, the data is recovered as it is at least part of

the message, and this data must be written to the local memory. Note that the receiving via a bus is done by the "communication object" and the message is recovered in part because of the first data transfer object which allows the message to be read from the buffer by setting the read-enabled flag. See column 2, lines 56-67.

46. Referring to claim 24, Tamura in view of Morikawa has taught a peer vector machine as described in claim 22. Tamura in view of Morikawa has further taught:

a) a registry coupled to the processor and operable to store object data. See Fig.4 of Tamura, and note that the objects 400 and 430 are software objects that include instructions.

b) wherein the processor is operable to execute an object factory and to generate the first and second data-transfer objects and the communication object from the object data under the control of the object factory. All processors execute programs. The program (object factory) will dictate when data needs to be transmitted and received. That is, when the program calls for data to be transmitted, received, and driven onto a bus, the first and second data transfer objects (Tamura, Fig.4, components 400 and 430) will be invoked/generated so that data may be transmitted and the object factory will be generated to that the data may be driven onto a bus.

47. Referring to claim 51, Tamura has taught a method comprising:

a) publishing data with an application running on a processor. See column 8, lines 63-64, and note that data of array A is published (i.e., generated). It should be noted that processors execute applications, which cause processing to occur.

b) loading the published data into a buffer with a first data-transfer object running on the processor. See Fig.6, and note the first data transfer object (transmit object (Fig.4, component 400)) loads the published data into the buffer.

c) retrieving the published data from the buffer with a second data-transfer object running on the processor. See Fig.6, and note that the second data transfer object (receive object (Fig.4, component 430)) retrieves the data from the buffer.

d) driving the retrieved published data onto a bus with a communication object running on the processor. Clearly, the data is retrieved via a bus and then sent somewhere. The object which sends the data somewhere is the communication object.

e) Tamura has not taught receiving the published data from the bus and processing the published data with a pipeline accelerator. However, Morikawa has taught the concept of a pipeline coprocessor (Fig.4, component 202, and the bottom of Fig.5 and Fig.6 show that the coprocessor is pipelined) that receives data from a processor via a buffer/bus, processes the data, and then passes it back via a buffer. As disclosed in Morikawa in column 1, lines 15-19, coprocessors execute special instructions at high speed, and therefore, a processor, which is not especially equipped to handle such instructions at high speed, will pass off instructions/data to the coprocessor so that it may perform the operation at a more rapid pace. This clearly speeds up the system, and as a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Tamura such that the portion of the system receiving the data is a pipeline accelerator (coprocessor) which performs operations at high speed for the processor.

48. Claims 38, 44, 46, 50, and 53-54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nakagoshi.

49. Referring to claim 38, Nakagoshi has taught a method as described in claim 37. Nakagoshi has not taught that the data comprises publishing the data with a thread of the

application. However, Official Notice is taken that multithreaded processors and their advantages are well known and accepted in the art. Specifically, it is known to divide up a program into threads in order to increase efficiency by reducing stall time. With multiple threads, the system may switch to a second thread when a first thread stalls, thereby hiding the stall time require by the first thread. Essentially, the processor is kept busy as often as possible with multithreading. As a result, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that the processor executes a thread of the application and publishes data with the thread.

50. Referring to claim 44, Nakagoshi has taught a method as described in claim 37. Nakagoshi has further taught receiving the message and processing the data in the message (Fig.1, the P00-P33 elements), but has not taught processing the data with a pipeline accelerator. However, Official Notice is taken that pipelines and their advantages are well known and accepted in the art. A pipeline allows for the overlapping of execution, instead of serial execution, thereby speeding up the processor and increasing throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that the processing elements (P00-P33) include pipelines (pipeline accelerators) for accelerating execution.

51. Referring to claim 46, Nakagoshi has taught a method as described in claim 45. Nakagoshi has not taught that processing the unloaded data comprises processing the unloaded data with a thread of the application corresponding to the destination. However, Official Notice is taken that multithreaded processors and their advantages are well known and accepted in the art. Specifically, it is known to divide up a program into threads in order to increase efficiency

by reducing stall time. With multiple threads, the system may switch to a second thread when a first thread stalls, thereby hiding the stall time require by the first thread. Essentially, the processor is kept busy as often as possible with multithreading. As a result, in order to increase efficiency, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that the processor processes the unloaded data with a thread.

52. Referring to claim 50, Nakagoshi has taught a method as described in claim 45. Nakagoshi has further taught generating the message header and the message (Fig.1-2, Fig.5, and column 7, lines 54-55), but has not taught generating with a pipeline accelerator. However, Official Notice is taken that pipelines and their advantages are well known and accepted in the art. A pipeline allows for the overlapping of execution, instead of serial execution, thereby speeding up the processor and increasing throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that the processing elements (P00-P33) include pipelines (pipeline accelerators) for accelerating execution. The portion generating the message w/ header may be thought of as part of the pipeline accelerator.

53. Referring to claim 53, Nakagoshi has taught a method comprising:

a) generating with a message header that includes a destination of data. See Fig.5, component 501.

b) generating a message that includes a header and the data. See Fig.5.

c) driving the message onto a bus. See Fig.1, Fig.9, and column 11, lines 1-24. Note that nodes pass messages between each other, and to do so, messages must be passed on a bus.

d) receiving the message from the bus with a communication object. Data must be received on a bus before anything can be done with it. Therefore, the inherently existing object receiving data from the bus is a communication object.

e) loading the received data into a buffer with a first data-transfer object running on a processor, the buffer being identified by the destination. See Fig.9. Note that data is received by an element and loaded into a buffer (FIFO). Clearly, the hardware must be controlled by some software (i.e., some command to cause the loading to occur). The hardware and software combination that loads the data is the first data transfer object.

f) unloading the data from the buffer with a second data-transfer object running on the processor. See Fig.9, and column 11, lines 1-24. Note that data is later unloaded from the FIFO for either processing or further routing. Again, in order to unload the data, a command from the processor must be given to unload the data. The hardware/software which causes the unloading is a second data transfer object.

g) processing the unloaded data with an application running on the processor and identified by the destination. A processor which receives the data (as shown in Fig.1 and Fig.9) will process the data in some form. Otherwise, there'd be no point in passing data. And, the processor to do the processing will be the one that the message was passed to (the destination).

f) Nakagoshi has not taught that the generation of a message header and message and the driving of the message onto a bus is performed by a pipeline accelerator. However, Official Notice is taken that pipelines and their advantages are well known and accepted in the art. A pipeline allows for the overlapping of execution, instead of serial execution, thereby speeding up the processor and increasing throughput. As a result, it would have been obvious to one of ordinary

skill in the art at the time of the invention to modify Nakagoshi such that the processing elements (P00-P33) include pipelines (pipeline accelerators) for accelerating execution. The portion generating the message w/ header and causing the driving of the message may be thought of as at least part of the pipeline accelerator.

54. Referring to claim 54, Nakagoshi has taught a method as described in claim 53. Nakagoshi has further taught recovering the data from the message with the first data transfer object. Since the message includes the data and a header, the data would have to be extracted. See Fig.9 and column 11, lines 1-24. The portion doing the extraction is the "first data transfer object".

55. Claim 52 is rejected under 35 U.S.C. 103(a) as being unpatentable over Tamura in view of Morikawa and further in view of Microsoft.

56. Referring to claim 52, Tamura in view of Morikawa has taught a method as described in claim 51.

a) While Tamura has taught generating a message (a message is simply the data transmitted and received), Tamura in view of Morikawa has not taught generating a message that includes a header and the published data with the second data-transfer object. However, Microsoft has taught that a header is an information structure that precedes and identifies the information that follows, such as a block of bytes in communications (i.e., body of a message). Headers are common in message passing and they typically include parity bits (or other error detection bits), to ensure that the data received is error-free, and the length of the data that follows so that it can be detected whether or not the entire message has been received. As a result, it would have been

obvious to one of ordinary skill in the art at the time of the invention to modify Tamura in view of Morikawa to attach a header, as taught by Microsoft, to the data sent in order to detect errors and length of the data. And, the second object, by retrieving the message, generates the message for additional processing on the receiving end.

b) Tamura in view of Morikawa has further taught that driving the data onto the bus comprises driving the message onto the bus with the communication object. The data is retrieved and then sent somewhere via a bus. The object which sends the data somewhere is the communication object.

c) Tamura in view of Morikawa has further taught that receiving and processing the published data comprises receiving the message and recovering the published data from the message with the pipeline accelerator. Again, recall Fig.4 of Morikawa where a message is received and the data in the message is processed by the accelerator (i.e., coprocessor).

Response to Arguments

57. Applicant's arguments filed on January 25, 2007, have been fully considered but they are not persuasive.

58. Applicant argues the novelty/rejection of claim 1 on page 26 of the amendment, in substance that:

> "Claim 1 recites a processor operable to load data into a buffer and to retrieve the data from the buffer...In contrast, Tamura does not disclose a processor operable to load data into a buffer and to retrieve the data from the buffer. Referring, e.g., to Tamura's FIG. 4, a first processor 410 is operable only to load data into a buffer 442a, and a second processor 420 is operable only to retrieve data from the buffer. The Examiner's position that the processors 420 and 430 effectively form a single processor is not supported by Tamura. Tamura discloses the processors 420 and 430 as separate processors that are each coupled to an external storage device 440 via respective busses, not via a single bus as one would expect for a single processor coupled to a memory."

These arguments are not found persuasive for the following reasons:

a) The examiner asserts that applicant is reading the claim too narrowly. The claim calls for a processor operable to perform the claimed steps. Tamura has taught a multiprocessor operable to perform the claimed steps. While a processor is not necessarily a multiprocessor, a multiprocessor is always a processor (more specifically, it is a specific type of processor). Whether or not a single bus or multiple buses couple the multiprocessor rot memory is irrelevant as applicant has no claim language pertaining to a bus structure in the claims.

59. Applicant's arguments with respect to claim 37 on pages 26-27 have been considered but are moot in view of the new ground(s) of rejection.

60. Applicant argues the novelty/rejection of claim 19 on page 29 of the amendment, in substance that:

> "Tamura does not disclose a processor operable to load data into a buffer under the control of a first data-transfer object and to retrieve the data from the buffer under the control of a second data-transfer object as discussed above in support of the patentability of claim 1."

These arguments are not found persuasive for the following reasons:

a) Applicant is merely asserting that Tamura doesn't teach specific features but does not disclose why these features are not taught. The examiner asserts that Tamura does disclose such features and the appropriate passages have been included in the rejections above.

61. Applicant's arguments with respect to claim 53 on pages 30-31 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

62. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

DJH
David J. Huisman
June 27, 2007

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100